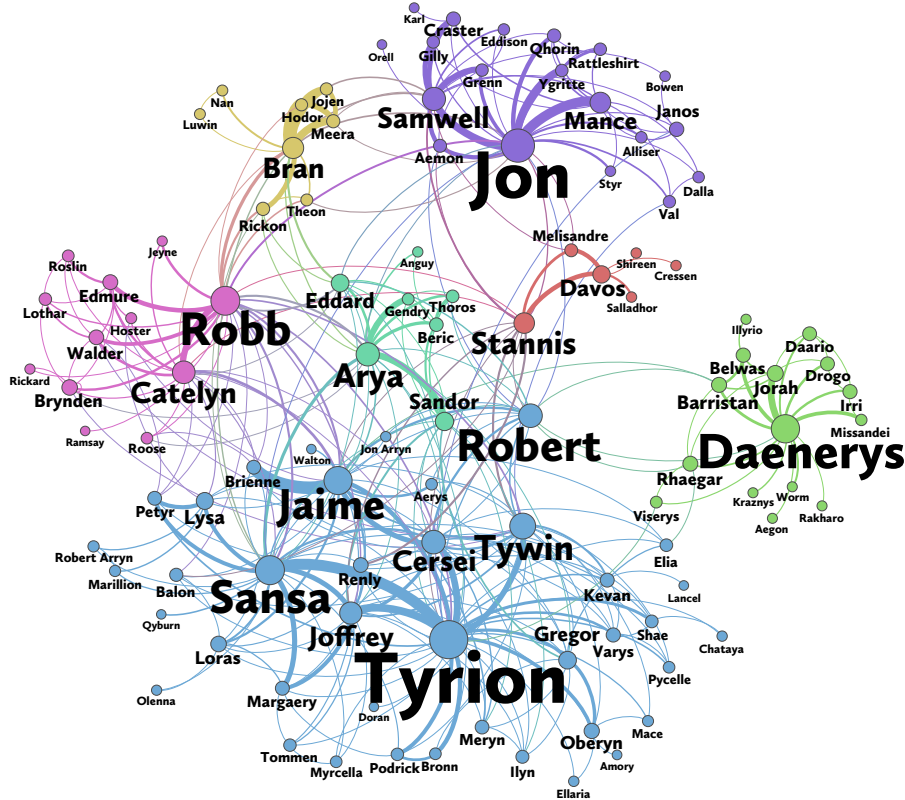


Exercises Day 1-1: Drawing Networks in R

Oldenburg, 11/01/2018

Part 1: Visualizing the network

In these exercises, we will analyse a social network of book series *A Song of Ice and Fire*, on which the popular TV show *Game of Thrones* is based. This network was [recently published in Math Horizons Magazine](#) (Beveridge & Shan, 2016), and is based on the third book, *A Storm of Swords*, on which the third and fourth season of the TV series are based (so no spoilers on the current season ;)). More information, including the data, is available [online](#). The network published is the following:



For these exercises, please refer to the [Network analysis cookbook](#), the [course materials](#), and the [qgraph manual](#).

Exercise 1 Look through the paper to get an idea of what this network represents. What do the nodes and edges represent?

The data as published online can be loaded in R as follows:

```
Data <- read.csv("stormofswords.csv")
```

Exercise 1 Look at the data in RStudio using the ‘View’ function. This matrix encodes a network. Can you figure out how? What do the rows stand for and what do the columns stand for?

This structure is known as an *edgelist* encoding a network, which can also be used as input for qgraph:

```
library("qgraph")
qgraph(Data, directed = FALSE)
```

Exercise 2 Why did I had to use the `directed` argument? What does it do?

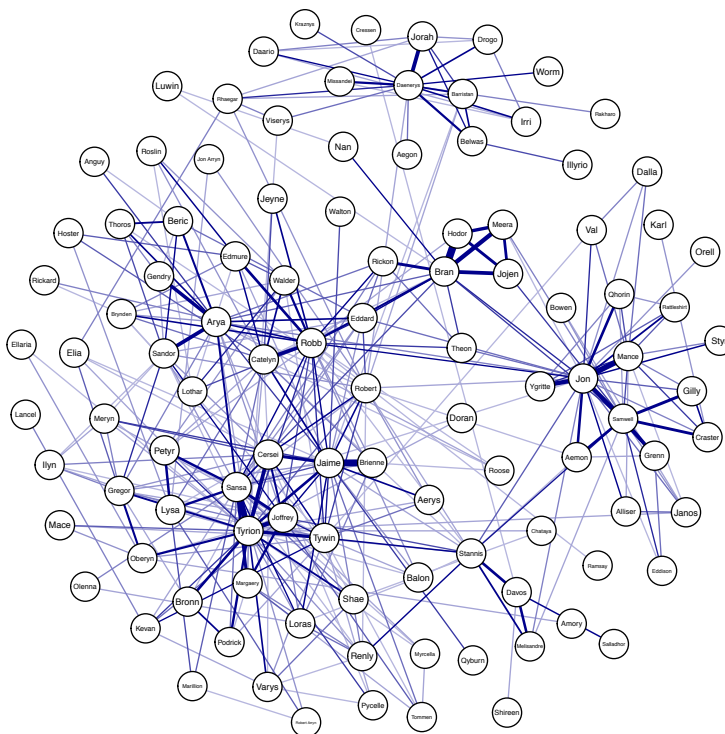
Now load the following dataset:

```
Data2 <- read.csv("stormofwords_wmat.csv")
```

This dataset represents the same network, but in a different way. This representation is called a *weights matrix*.

Exercise 3 Investigate the new data using `View(Data2)`. Can you figure out how this weights matrix encodes a network. Use `Data2` as input to `qgraph()`. Do you obtain the same network? Do you still need to use the `directed` argument?

The plotted network is plotted using a circular layout. This circular layout, however, is hard to interpret and read in such networks with many nodes. In addition, I like to plot edges with a different color than green when they represent values that can only be positive. Finally, the nodes are very large and we might want to make them smaller:



Exercise 4 Recreate the plot above, changing the layout to a spring layout, the edge color to `"darkblue"` and the node size to 3. Look at the `qgraph` help page to figure out the commands needed (`?qgraph`). Note: your computer might generate a different spring layout (nodes placed on different locations).

`qgraph` uses three arguments that need to be known to interpret a network: `minimum`, `cut`, and `maximum`. These are set automatically, and can be shown using the argument `details = TRUE`.

Exercise 5 What values did `qgraph` set to `cut` and `maximum`? Note: `minimum` is always set to 0 by default and not shown with `details = TRUE` unless it differs from 0.

The `minimum` argument can be used to *hide* edges with an absolute (negative edges are treated as positive) weight under some value. Note that these edges are only visually hidden, not removed in further analyses (which can be done using the `threshold` argument). This argument is useful when plotting dense graphs

(e.g., correlation networks) but *not* recommended in the networks estimated in this workshop.

Exercise 6 Set the `minimum` argument to 1, 5 and 10 while using a spring layout. How does the network change? Do the same using the `threshold` argument. Can you explain why the layout remains the same using `minimum` but changes using `threshold`?

Edges drawn in `qgraph` are drawn wider and more saturated the stronger the absolute edge weight is. In large networks, it might be useful to split the scaling of width and color. This is what the `cut` argument does: edges with an absolute edge weight under the cutoff will be drawn of the smallest width and vary only in color. Edges with an absolute edge weight over the cutoff value will always be drawn fully saturated (green or red by default), and will be drawn thicker the stronger they are. You can disable this behavior by setting `cut = 0`, which will ensure that edges all scale in color and width. `qgraph` automatically sets a cutoff value when there are 20 or more nodes in the network. We recommend setting `cut = 0` unless you specifically want this behavior.

Exercise 7 Set the `cut` argument to 50, 10 and 1 while using a spring layout. How does the network change? Now set the `cut` argument to 0. What happened?

The scaling of edges is chosen based on the strongest absolute edge in the network. This is because networks highly differ in their weights. This social network is based on the number of interactions. The characters “Bran” and “Hodor” interacted the most (96 times), leading to a strongest edge weight of 96. A network based on (partial) correlations, however, can never have a stronger edge weight than 1 (and usually features weaker edges). The `maximum` argument can be used to overwrite the “largest edge” to which the color and width of edges scale to. Its value is treated as the weight of an invisible edge, and is automatically set to the largest edge weight in the network. Setting `maximum` higher will make edges scale to that value instead of the strongest edge.

Exercise 8 Set the `maximum` argument to 200. What happened? Now set `maximum` back to its default and subsequently to 10. Why doesn’t `maximum = 10` change the network from its default value, but `maximum = 200` does?

Exercise 9 What would be a good setting for `maximum` when drawing networks based on (partial) correlations?

Part 2: Analyzing the network

Now that we can visualize the network, we might wish to analyze it further. To do this, we first need to store the network:

```
Graph <- qgraph(Data, directed = FALSE, layout = "spring")
```

Now we can obtain the weights matrix for further analysis (this is how I obtained Data2):

```
WeightMat <- getWmat(Graph)
```

Its column names tells us the labels of each node:

```
Labels <- colnames(WeightMat)
```

Now we can see that Jon Snow is node 32:

```
which(Labels == "Jon")
```

```
## [1] 32
```

and Daenerys is node 17:

```
which(Labels == "Daenerys")
```

```
## [1] 17
```

One option is to analyze how important or *central* nodes are in the network. A description of how these measures can be computed can be found in Opsahl, Agneessens, and Skvoretz (2010). These can be analyzed using the `centrality` function:

```
Centrality <- centrality(Graph, all.shortest.paths = TRUE)
```

Node strength (also called degree) sums the connected edge weights to a node. The node strength of Jon and Daenerys are:

```
Centrality$OutDegree[c(17, 32)]
```

```
## Daenerys      Jon
##      232      442
```

Closeness computes how “close” two nodes are together in the network:

```
Centrality$Closeness[c(17, 32)]
```

```
## Daenerys      Jon
## 0.02612782 0.04305410
```

Finally, *Betweenness* computes how often one node is featured in the most efficient (shortest) paths between other nodes:

```
Centrality$Betweenness[c(17, 32)]
```

```
## Daenerys      Jon
##      2200      2838
```

Based on these results, Jon seems more important in this book.

Exercise 10 Compare the centrality of the character named “Tyrion” to those of Jon and Daenerys. ■

We can also look at how “close” Jon and Daenerys are:

```
Centrality$ShortestPathLengths[17,32]
## [1] 0.3969188
```

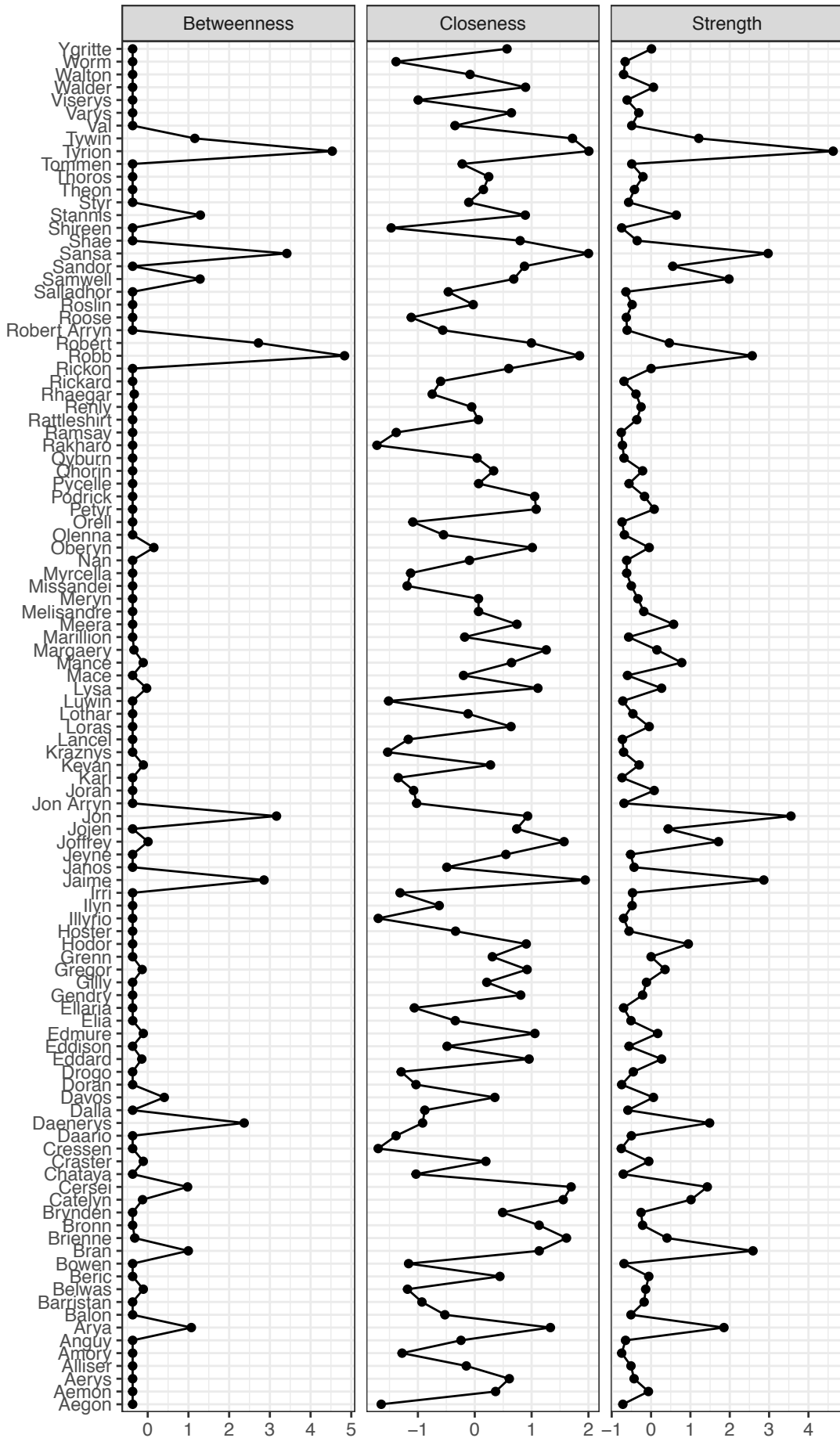
The following characters connect Jon and Daenerys:

```
Labels[Centrality$ShortestPaths[[17,32]][[1]]]
## [1] "Daenerys" "Robert" "Jaime" "Robb" "Jon"
```

Exercise 11 Via which characters is Tyrion connected to Jon and Daenerys? ■

We can plot the centrality indices using the `centralityPlot` function:

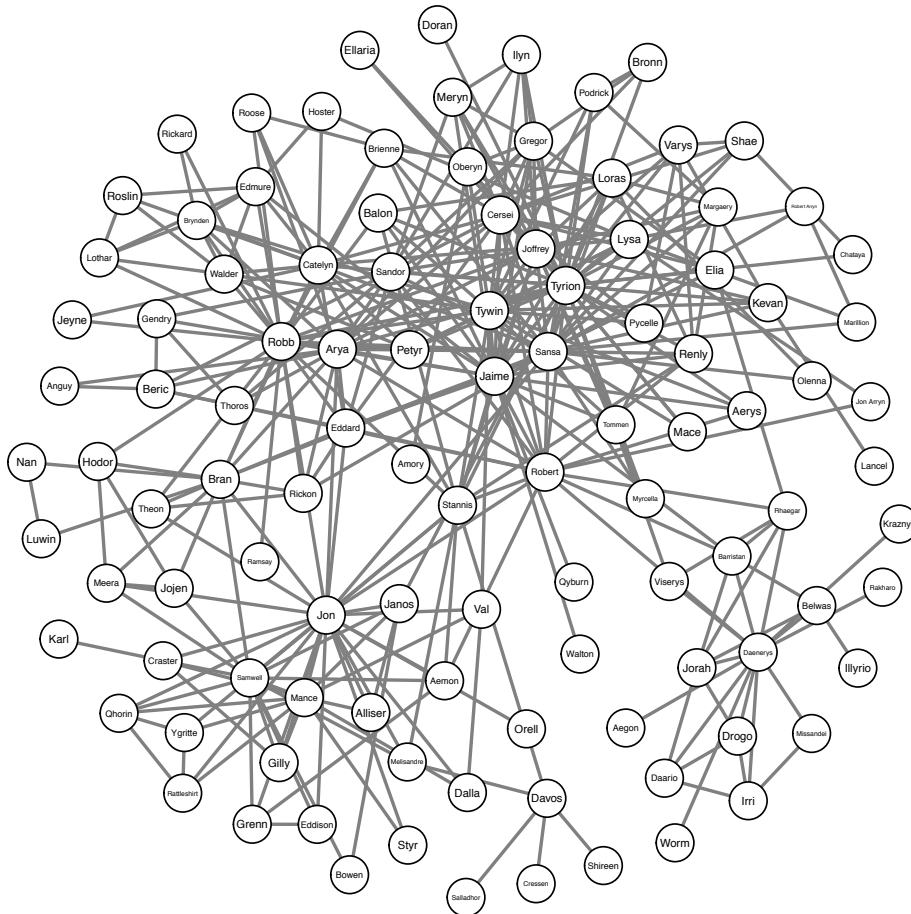
```
centralityPlot(Graph)
## Note: z-scores are shown on x-axis rather than raw centrality indices.
```



Exercise 12 What character would you conclude is the most central?

Many network analysts prefer to analyze an unweighted network rather than a weighted network. We can strip the weights by setting all edge weights to 1 or by not using the weights in the edgelist:

```
Graph_unweighted <- qgraph(Data[,1:2], directed=FALSE, repulsion = 0.7)
```



Exercise 13 What happened when I set the `repulsion` argument lower than its default of 1?

Exercise 14 Recreate the centrality plot using the unweighted network. Does your conclusion on the most central character change?

One common analysis of such network structures is to investigate if the network exhibits a “small-world” structure. A network has a small-world structure if it is (a) structured such that it forms clusters: my friends also talk to each other, and (b) most nodes can reach each other with only a few steps (related to the idea of “six degrees of separation”). A small-world arises due to networks being structured in clusters (my friends talk to each other) but also featuring some random connections (e.g., due to attending international

conferences, you now also talk to people from all over the world). For more details, see the classical paper by Watts and Strogatz (1998). We can use the `smallworldIndex` to obtain a “small-world index”. This can only be computed from unweighted networks. If the index is above 3, we can conclude the network features a small world.

Exercise 15 Does the unweighted network feature a small world?

References

- Beveridge, A., & Shan, J. (2016). Network of thrones. *Math Horizons*, 23(4), 18–22.
- Opsahl, T., Agneessens, F., & Skvoretz, J. (2010). Node centrality in weighted networks: Generalizing degree and shortest paths. *Social networks*, 32(3), 245–251.
- Watts, D., & Strogatz, S. (1998). Collective dynamics of ‘small-world’ networks. *Nature*, 393(6684), 440–442.